|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[유헬스2\_10주\_실습과제]** | **학번:** |  | **이름:** |  |

* 인터페이스
* 인터페이스 정의

public **interface** OperateCar {

void start(); //추상 메소드, abstract 생략가능

void stop();

void setSpeed(int speed);

}

* 인터페이스 구현

public class AutoCar implements OperateCar {

public void start() {

System.out.println("자동차가 출발합니다.");

}

public void stop() {

System.out.println("자동차가 정지합니다.");

}

public void setSpeed(int speed) {

System.out.println("자동차가 속도를 " + speed + "km/h로 바꿉니다.");

}

}

* 인터페이스를 구현한 클래스 테스트

public class AutoCarTest {

public static void main(String[] args) {

OperateCar obj = new AutoCar();

obj.start();

obj.setSpeed(30);

obj.stop();

}

}

* 무명 클래스

interface RemoteControl {

void turnOn();

void turnOff();

}

public class AnonymousClassTest {

public static void main(String args[]) {

**RemoteControl ac = new RemoteControl()** { **// 무명 클래스 정의**

public void turnOn() {

System.*out.println("TV turnOn()"); }*

*public void turnOff() {*

*System.out.println("TV turnOff()"); }*

*}****;*** **// ; 반드시 기입**

*ac.turnOn();*

*ac.turnOff(); }*

*}*

|  |
| --- |
| **[실행결과]** |

* 함수 인터페이스와 람다식

//람다식을 위한 함수인터페이스 임을 나타내는 anotation

@FunctionalInterface

interface Func1{

int calc(int a, int b);

}

@FunctionalInterface

interface Func2{

void sayHello();

}

public class ArrayTest {

public static void main(String[] args) {

Func1 sub = (int a, int b) -> a - b;

Func1 add2 = (int a, int b) -> { return a + b; };

Func2 hello = () -> System.*out.println("Hello Lambda!");*

*// or {System.out.println("Hello Lambda!"); }*

int result=sub.calc(3,4)+add2.calc(5,2);

System.*out.println("result = " + result);*

hello.sayHello();

}

}

|  |
| --- |
| **[실행결과]** |

* 내용 점검 문제

1. 다음 주어진 인터페이스와 클래스를 보고 물음에 답하시오.

|  |
| --- |
| **public** interface Mammal { //Mammal 인터페이스 정의  **void** walk();  void giveBirth();  } |

|  |
| --- |
| **public** **class** Person implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "사람";  public void walk() {  System.*out*.println(*TYPE* + "은 두 발로 걷습니다.");  }  **public** **void** giveBirth() {  System.*out*.println(*TYPE* + "이 아기를 낳습니다.");  }  } |

|  |
| --- |
| **public** **class** Whale implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "고래";  **public** **void** walk() {  System.*out*.println(*TYPE* + "는 물에서 수영합니다.");  }  public void giveBirth() {  System.*out*.println(*TYPE* + "는 아기를 낳습니다.");  }  } |

1. 위 프로그램에서 빈 부분을 채우시오..
2. 다음 문장이 오류가 발생하는 이유를 설명하시오.

Mammal mal = **new** Mammal();

인터페이스는 객체생성이 불가능하다.(자료타입으로는 가능)

1. 다음 소스의 결과를 기술하시오.

Person p = **new** Person();

p.walk();

p.giveBirth();

Whale w = **new** Whale();

w.walk();

w.giveBirth();

사람은 두 발로 걷습니다.

사람이 아기를 낳습니다.

고래는 물에서 수영합니다.

고래는 아기를 낳습니다.

1. 다음 소스의 결과를 기술하시오.

Mammal m[] = {**new** Person(), **new** Whale()};

m[0].walk();

m[0].giveBirth();

m[1].walk();

m[1].giveBirth();

사람은 두 발로 걷습니다.

사람이 아기를 낳습니다.

고래는 물에서 수영합니다.

고래는 아기를 낳습니다.

1. 다음과 같은 인터페이스 클래스가 있다고 하자. MyClass에서 구현하여야 하는 메소드는 어떤 것인가?

interface A {

public float mA(int a);

}

interface B extends A {

public int mB(int a);

public Object mBB(int a);

}

class C {

public void mC(int a) {

System.out.println("안녕하세요");

}

}

public class MyClass extends C implements B {

}

mA, mB, mBB 를 구현해야한다

1. 다음의 인터페이스 선언과 사용에서 잘못된 점을 모두 지적하라.

public interface Edible {

boolean amount;

final int TYPE=10;

public void eat() { };

};

public class Sandwitch extends Edible {

public void eat() { }

}

extends -> implements // 인터페이스를 구현할때는 implements를 쓴다.

boolean amount; -> final boolean amount=false //인터페이스는 필드가 없다 그러므로 지우던지 상수로 정의 해줘야 한다.

public void eat() { }; -> public void eat(); // 추상메소드를 쓸때는 몸체가 없어야 하므로 중괄호가 빠져야 한다.

1. 다음과 같은 조건을 만족하는 프로그램을 작성하라.

(a) 인터페이스 GraphicsObject에서는 int getArea( )과 void draw( )가 선언된다.

(b) GraphicsObject을 구현하는 “Rectangle”클래스를 작성한다.

(c) Rectangle 클래스는 2개의 private 정수 필드인 length와 width을 가진다.

(d) setDimensions (int l, int w) 메소드는 length와 width의 값을 설정한다.

|  |
| --- |
| **[프로그램 소스]**  **public** **interface** GraphicsObject {  **int** getArea();  **void** draw();  }  **public** **class** Rectangle **implements** GraphicsObject {  **private** **int** length;  **private** **int** width;  **public** **int** getArea() {  **return** length \* width;  }  **public** **void** draw() {  **for** (**int** i = 0; i < length; i++) {  **for** (**int** j = 0; j < width; j++) {  System.***out***.print("\*");  }  System.***out***.println();  }  }  **public** **void** setDimensions(**int** l, **int** w) {  **this**.length = l;  **this**.width = w;  }  } |

1. 비디오 플레이어는 play와 stop라는 조작을 할 수 있다. 따라서 인터페이스 controllable을 아래와 같이 정의할 수 있다. controllable 인터페이스를 구현하는 VideoPlayer 클래스를 Test 클래스의 main() 안에 무명 내부 클래스로 작성하고 테스트하여 보자.

**public** **interface** controllable {

**void** play();

**void** stop();

}

|  |
| --- |
| **[프로그램 소스]**  **public** **class** Test {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  controllable VideoPlayer = **new** controllable() {  **public** **void** play() {  System.***out***.println("VideoPlayer play");  }  **public** **void** stop() {  System.***out***.println("VideoPlayer stop");  }  };  VideoPlayer.play();  VideoPlayer.stop();  }  } |
| **[실행결과]** |

* 프로그램 과제

1. 다음과 같은 메소드로 구성되는 인터페이스 Dim을 정의하시오

* void print()
* void init(int size);

1. Dim 인터페이스를 구현하는 클래스 IntDim을 정의하시오

* 필드 : int 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
  + print() 는 배열 원소를 출력
  + init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. Dim 인터페이스를 구현하는 클래스 DoubleDim을 정의하시오

* 필드 : double 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
  + print() 는 배열 원소를 출력
  + init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. Dim 인터페이스를 구현하는 클래스 CharDim을 정의하시오

* 필드 : char 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
* print() 는 배열 원소를 출력
* init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. 실행 후 결과를 확인하세요

**import** java.util.\*;

**public** **class** Array\_EX\_Test {

**public** **static** **void** main(String[] args) {

Scanner in=**new** Scanner(System.*in*);

Dim array = **null**; //객체 참조 변수 초기화

System.*out*.println("1: 정수형 배열, 2:실수형 배열 3. 문자형 배열" );

**int** menu=in.nextInt();

**if**(menu >= 4)

{

*messAge*("잘못 선택하셨습니다.");

*messAge*("다시 입력하세요.");

menu=in.nextInt();

}

//해당 메뉴를 처리할 수 있는 객체를 생성하여 참조변수 array에 저장

**switch**(menu){

**case** 1: array = **new** Intdim(); //다형성 Dim타입의 변수로 Intdim객체 저장

**break**;

**case** 2: array = **new** DoubleDim(); //다형성 Dim타입의 변수로 Doubledim객체 저장

**break**;

**case** 3: array = **new** CharDim();

**break**;

}

*messAge*("생성할 배열 크기를 입력하세요");

**int** size = in.nextInt();

array.init(size); //다형성, array가 가리키는 객체의 메소드 호출하여 실행

array.print(); //다형성, array가 가리키는 객체의 메소드 호출하여 실행

*messAge*("프로그램을 종료합니다");

}

**public** **static** **void** messAge(String ms){

System.*out*.println(ms);

}

}

|  |
| --- |
| **[프로그램 소스]**  **public** **interface** Dim {  **void** print();  **void** init(**int** size);  }  **public** **class** IntDim **implements** Dim {  **int** arr[];  **public** **void** print() {  **for** (**int** temp : arr) {  System.***out***.println(temp);  }  }  **public** **void** init(**int** size) {  arr = **new** **int**[size];  **for** (**int** i = 0; i < size; i++) {  arr[i] = (**int**) (Math.*random*() \* 99 + 1);  }  }  }  **public** **class** DoubleDim **implements** Dim {  **double** arr[];  **public** **void** print() {  **for** (**double** temp : arr) {  System.***out***.println(“%5.2f”,temp);  }  }  **public** **void** init(**int** size) {  arr = **new** **double**[size];  **for** (**int** i = 0; i < size; i++) {  arr[i] = (Math.*random*() \* 99 + 1);  }  }  }  **public** **class** CharDim **implements** Dim {  **char** arr[];  **public** **void** print() {  **for** (**char** temp : arr) {  System.***out***.println(temp);  }  }  **public** **void** init(**int** size) {  arr = **new** **char**[size];  **for** (**int** i = 0; i < size; i++) {  arr[i] = (**char**) ((**int**) (Math.*random*() \* 99 + 1));  }  }  } |
| **[실행결과]** |

1. 제시된 조건대로 프로그램을 작성하시오

* 함수형 인터페이스 FPlay

- void play(String str) //매개변수로 받은 문자열을 출력, “OOO을 재생 합니다”

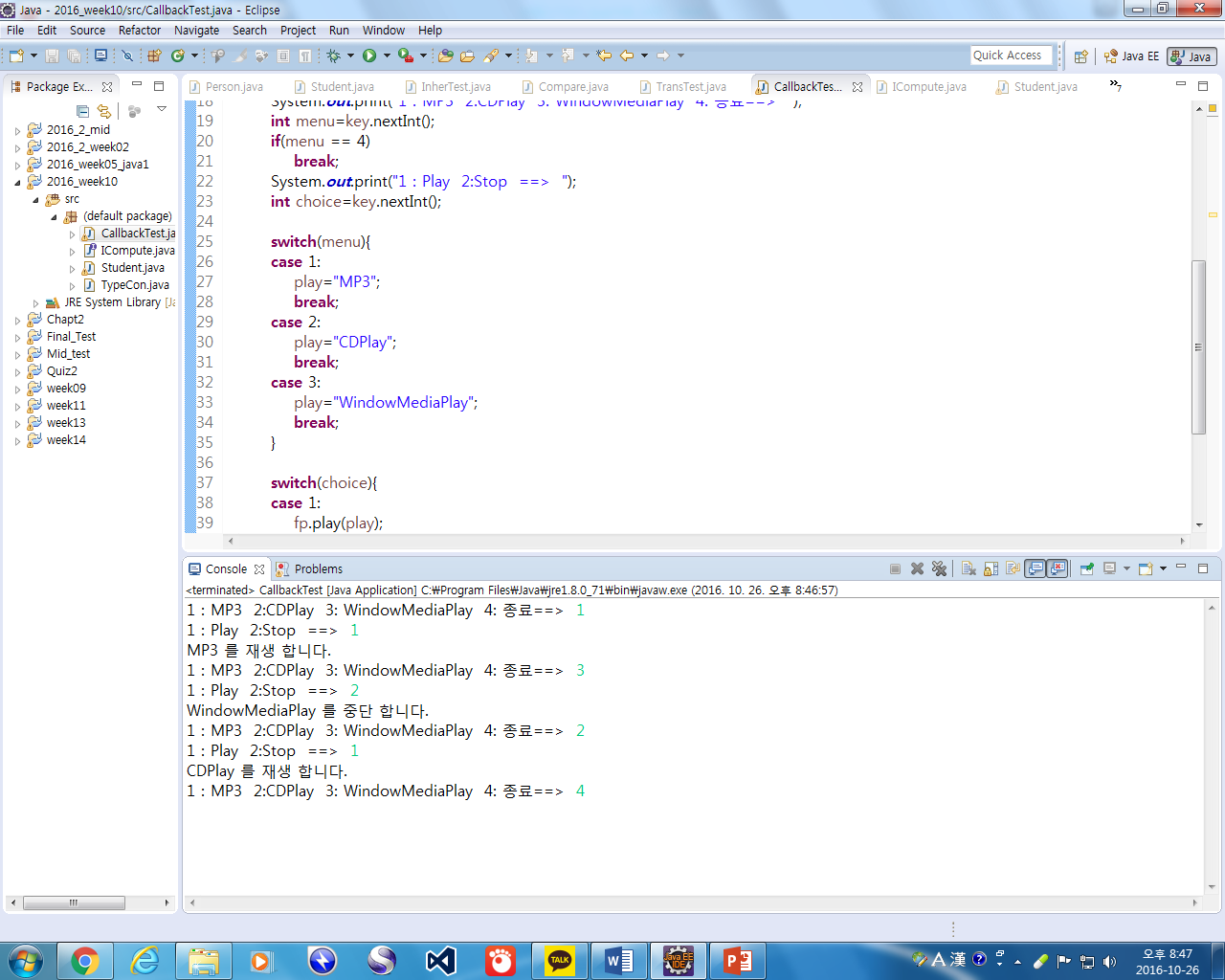
* 함수형 인터페이스 FStop

- void stop(String str) //매개변수로 받은 문자열 출력, “OOO을 중단합니다”

* InterfaceTest 클래스

- main()

* 함수형 인터페이스에 대한 람다식 정의
* 메뉴에 따라 play(), stop() 메소드 실행



|  |
| --- |
| **[프로그램 소스]**  **public** **interface** FPlay {  **void** play(String str);  }  **public** **interface** FStop {  **void** stop(String str);  }  **import** java.util.\*;  **public** **class** InterfaceTest {  **public** **static** **void** main(String[] args) {  Scanner scan = **new** Scanner(System.***in***);  FPlay FP = (String str) -> {  System.***out***.println(str + "을 재생합니다.");  };  FStop FS = (String str) -> {  System.***out***.println(str + "을 중단합니다.");  };  **boolean** boo = **true**;  **while** (boo) {  System.***out***.print("1:MP3 2:CDPlay 3:WindowMediaPlay 4:종료==> ");  **switch** (scan.nextInt()) {  **case** 1:  System.***out***.print("1:Play 2:Stop ==>");  **switch** (scan.nextInt()) {  **case** 1:  FP.play("MP3");  **break**;  **case** 2:  FS.stop("MP3");  }  **break**;  **case** 2:  System.***out***.print("1:Play 2:Stop ==>");  **switch** (scan.nextInt()) {  **case** 1:  FP.play("CDPlay");  **break**;  **case** 2:  FS.stop("CDPlay");  }  **break**;  **case** 3:  System.***out***.print("1:Play 2:Stop ==>");  **switch** (scan.nextInt()) {  **case** 1:  FP.play("WindowMediaPlay");  **break**;  **case** 2:  FS.stop("WindowMediaPlay");  }  **break**;  **case** 4:  System.***out***.println("종료합니다");  boo = **false**;  }  }  }  } |
| **[실행결과]** |